CLAIMS

1. A cryptographic method in an electronic component during which a modular exponentiation of the type x^d is performed, with d an integer exponent of m+1 bits, by scanning the bits of d from left to right in a loop indexed by i varying from m to 0 and calculating and storing in an accumulator (R0), at each turn of rank i, an updated partial result equal to x^b(i), b(i) being the m-i+1 most significant bits of the exponent d ($b(i) = d_{m->i}$),

the method being characterised in that, at the end of a turn of rank i(j) (i = i(0)) chosen randomly, a randomisation step E1 is performed during which:

E1: a random number z (z = b(i(j)), $z = b(i(j)).2^{\tau}$, z = u) is subtracted from a part of the bits of d not yet used ($d_{i-1->0}$) in the method

then, after having used the bits of d modified by the randomisation step E1, a consolidation step E2 is performed during which:

E2: the result of the multiplication of the content of the accumulator (x^b(i)) by a number that is a function of x^z stored in a register (R1) is stored (R0 <- R1xR0) in the accumulator (R0).

2. Method according to the preceding claim, in which step E1 is repeated one or more times, at the end of various turns of rank i(j) (i = i(0), i = i(1), ...) chosen randomly between 0 and m.

3. Method according to the preceding claim, in which, at each turn i, it is decided randomly (ρ=1) whether or not step E1 is performed.

4. A cryptographic method according to one of claims 1 to 3, in which the number z (z=b(i(j)), z = b(i(j)).2^{τ}) is a function of the exponent d, in which, during the randomisation step, the result of the multiplication of the content of the accumulator (x^b(i)) by the content of the register (R1) is also stored (R1 <- R0xR1) in the said register (R1).

5. A method according to claim 4, in which the consolidation step E2 is performed after the last turn of rank i equal to 0.

6. A method according to the preceding claim, during which, during step E1, the number b(i) is subtracted from d.

7. A method according to claim 6, during which the following is effected:

Input: x, d = $(d_m, \ldots, d_0)_2$

Output: y = x^d mod N

R0 <- 1; R1 <-1; R2 <- x, i <- m

as long as i $\geq$ 0, do:

R0 <- R0xR0 mod N

if d_i = 1 then R0 <- R0xR2 mod N

ρ <- R{0, 1}

if ((ρ = 1) and $d_{i-1\to 0} \geq d_{m\to i}$ then

d $\leftarrow$ d – $d_{m->i}$

R1 <- R1xR0 mod N

end if

i <- i-1

end as long as

R0 <- R0xR1 mod N

return R0

8. A method according to claim 5, during which step E1 is modified as follows:

E1: a number equal to g.b(i) is subtracted from d, g being a positive integer; the current partial result (x^b(i)) is raised to the power of g and the result is stored in the register (R1).

9. A method according to the preceding claim, in which g is equal to 2^{τ}, τ being a random number chosen between 0 and T.

10. A method according to the preceding, in which the following is effected:

Input: x, d = $(d_m, \ldots, d_0)_2$

Output: y = x^d mod N

R0 <- 1; R1 <-1; R2 <- x, i <- m

as long as i $\geq$ 0, do:

R0 <- R0xR0 mod N

if d_i = 1 then R0 <- R0xR2 mod N

ρ <- R{0, 1}; τ <- R{0, ..., T}

if ((ρ = 1) and ($d_{i-1\to\tau} \geq d_{m\to i}$)) then

$d_{i-1\to\tau} \leftarrow d_{i-1\to\tau} - d_{m->i}$

R3 <- R0

as long as (τ > 0) do:

R3 <- R3^2 mod N; τ <- τ-1

end as long as

R1 <- R1xR3 mod N

```
            end if
            i <- i-1
        end as long as
        R0 <- R0xR1 mod N
    return R0
```

11. A method according to one of claims 1 to 4, in which the consolidation step E2 is performed at the end of the rank using the last bit of d modified during step E1.

12. A method according to claim 11, in the course of which, during step E1, the number b(i) is subtracted from the bits of d of rank i(j) - c(j) to i(j)-1, c(j) being an integer, and the content of the accumulator (x^b(i(j)) is stored in the register (R1).

13. A method according to the preceding claim, in the course of which, during the turn of rank i(j+1), it is chosen randomly to perform step E1 only if i(j+1) ≤ i(j)-c(j). (σ = 1 free semaphore).

14. A method according to claim 12 or 13, in which c(j) is equal to m-i(j)+1.

15. A method according to the preceding claim, during which the following steps are performed:

Input: x, $d = (d_m, \ldots, d_0)_2$

Output: y = x^d mod N

```
    R0 <- 1; R1 <-1; R2 <- x,
    i <- m; c <- -1; σ <- 1
    as long as i ≥ 0, do:
        R0 <- R0xR0 mod N
        if d_i = 1 then R0 <- R0xR2 mod N end if
```

```
            if (2i ≥ m+1) and (σ=1) then c <- m-i+1
                                    if not σ = 0
            end if
            ρ <- R{0, 1}
            ε <- ρ and (d_{i-1 ->i-c} ≥ d_{m→i}) and σ
            if ε = 1 then
                R1 <- R0; σ <- 0
                d_{i-1 -> i-c} <- d_{i-1 -> i-c} - d_{m->i}
            end if
            if c = 0 then
                R0 <- R0xR1 mod N; σ <- 1
            end if
            c <- c-1; i <- i-1
        end as long as
    return R0
```

16. A method according to claim 12 or 13, in which $c(j)$ is chosen randomly between $i(j)$ and $m-i(j)+1$.

17. A method according to the preceding claim, during which the following is effected:

```
Input:   x, d = (d_m,...,d_0)_2
Output:  y = x^d mod N
    R0 <- 1; R1 <- 1; R2 <- x,
    i <- m; c <- -1; σ <- 1
    as long as i ≥ 0, do:
        R0 <- R0xR0 mod N
        if d_i = 1 then R0 <- R0xR2 mod N
          if (2i ≥ m+1) and (σ = 1)
                     then c <- R{m-i+1, ..., i}
```

```
                    if not σ = 0
            ε <- ρ and (d_{i-1 ->i-c} ≥ d_{m->i}) and σ
            if ε = 1 then
                R1 <- R0; σ <- 0
                d_{i-1 -> i-c} <- d_{i-1 -> i-c} - d_{m->i}
            end if
            if c = 0 then
                R0 <- R0xR1 mod N; σ <- 1
            end if
            c <- c-1; i <- i-1
        end as long as
    return R0
```

18. A method according to one of claims 1 to 2, in which the number z is a number u (z = u) of v bits chosen randomly and independent of the exponent d.

19. A method according to the preceding claim, in which, during step E1, the number u is subtracted from a packet w of v bits of d.

20. A method according to the preceding claim, during which:

- if H(w-u) + 1 < H(w), it is chosen to perform a randomisation step E1,
- if H(w-u) + 1 > H(w), it is chosen not to perform step E1,
- if H(w-1) + 1 = H(w), it is chosen randomly to perform or not a randomisation step E1.

21. A method according to the preceding claim, during which the following is effected:

Input: x, $d = (d_m, \ldots, d_0)_2$

```
Parameters: v, k
Output:     y = x^d mod N
    R0 <- 1; R2 <- x; i <-m; L = {}
    as long as i ≥ 0, do:
        R0 <- R0xR0 mod N
        if d_i = 1 then R0 <- R0xR2 mod N end if
        if i = m mod ((m+1)/k)) then σ<-1 end if
        if σ = 1 and L = {} then
            s <- 0: u <- R {0, ..., 2^v-1};
            R1 = x^u mod N
        end if
        w <- d_{i->i-v+1}
        h <- H(w)
        if w ≥ u then Δ <- w-u; h_Δ <- 1 + H(Δ)
                if not h_Δ <- v+2
        end if
        ρ <- R{0, 1}
        if [(σ=0)∧(i-v+1≥0)] ∧
            [(h>h_Δ) or ((ρ=1) and (h=h_Δ))] then
            d_{i->i-v+1} <- Δ; L <- L ∪ {i-v+1}
        end if
        if (i ε L) then
        R0 <- R0xR1 mod N
        L <- L\{i}
        end if
        i <- i-1
    end as long as
return R0
```